<u>**Amendments to the Specification:**</u>

Please insert, on page 1, in the first line of the specification, following the title, the following new paragraph:

This application is a continuation of U.S. Patent Application Serial No. 10/339,571, filed January 10, 2003, which is a continuation of U.S. Patent Application Serial No. 09/984,525, filed October 30, 2001, which is a divisional of U.S. Patent Application Serial No. 09/223,299, filed December 30, 1998 (now U.S. Patent No. 6,338,132, issued January 8, 2002).

Please replace the paragraph at p. 4, line 20 with the following amended paragraph:

Fig. 7a shows two example instructions in the instruction format of [[Fig. 1]] <u>**Fig. 2**</u>

Please replace the paragraph beginning at p. 2, line 20 and extending to p. 3, line 8 with the following amended paragraph:

A typical instruction or micro-op includes a fixed number of bits, regardless of the information to be stored in each instruction. Figure 1 shows the format of an example instruction **100** in a conventional instruction format. The instruction includes an opcode field **101** and an immediate operand field **102**. In this particular instruction format the opcode field **101** has a bit length of X and the immediate operand field **102** has a bit length of Y. The fixed number of bits allocated to each instruction may be determined by the maximum number of bits [[exployed]] <u>employed</u> for each of the above listed programming elements. For example, the maximum number of bits for the opcode is known because the number of operations specified by the opcode is fixed. For example, if 6 bits are designated for the opcode field **101**, there can be maximum of 2^6 or 64 operations specified by the opcode, and each of these operations will be represented by a unique combination of the 6 bits. The fixed length of the bits (Y) allocated to the immediate operand field **102** is normally equal to the size of the maximum immediate operand that will occur in any instruction. For example, if the maximum immediate operand is 32 bits in length, each instruction will have a fixed length of 32 bits assigned for the immediate operand, regardless of whether 32 bits are employed for that particular instruction. Thus, each and every instruction formatted in the above described instruction format will have 6 bits allocated to the opcode field **101** and 32 bits allocated to the immediate operand field **102**, regardless of whether these bits are employed to express the function to be performed by the instruction.

Please replace the paragraph beginning at page 5, line 10 with the following replacement paragraph:

Embodiments of the present invention can be applied to any level of the program instruction, e.g., opcode, binary code, etc., thus, when using the term "instruction" throughout this specification it should be understood that it is not limited to any particular type of instruction. Likewise, throughout this specification the term "immediate operand" is used to describe an example type of instruction data that can be implemented using the present invention. Again, it should be understood that this is only an example, and any type of instruction operand can be used with the present invention.

Please replace the paragraph beginning at page 12, line 9 with the following rewritten paragraph:

Figure 6b shows the same portion of a program as Figure 6a, except that the instructions **620-630** are in the instruction format of Figure 2, after the immediate operand from ~~instruction **500**~~ instruction **600** has been compressed using forward scavenging. Again, the format of the example instructions **620-630** have a fixed number of bits allocated to the opcode fields **621, 631**, the control fields **622, 632** and the immediate operand fields **623, 633**. The fixed number of bits allocated to the immediate operand fields **623, 633** is 16 bits, half of the number of bits allocated to the immediate operand ~~fields **502, 512**~~ fields **602, 612** in the instructions **600-610** of Figure 6a. Forward scavenging compresses the 32 bit immediate operand in the following manner. The processor recognizes that the immediate operand for instruction **600** employs 32 bits to represent its value, and also that there is no immediate operand in instruction **610**. Therefore, the processor can store the immediate operand or a portion of the immediate operand of instruction **600** in the vacant immediate operand field **612** of instruction **610**. In this particular case, since 32 bits are employed to represent the immediate operand of instruction **600**, the processor accomplishes this compression by splitting the 32 bit immediate operand associated with instruction **600** in half. Fig. 6b shows the results of this compression, with 16 bits of the immediate operand stored in the immediate operand field **623** of the current instruction **620** and 16 bits of the immediate operand stored in the immediate operand field **633** of the next instruction **630**.